IT205 Course Project

Ayush Mangukia - 191IT211
Information Technology
National Institute of Technology Karnataka
Surathkal, India 575025
Email:ayushmangukia.191it211@nitk.edu.in

Mohammed Ibrahim - 191IT230
Information Technology
National Institute of Technology Karnataka
Surathkal, India 575025
Email:mibrahim.191it230@nitk.edu.in

Soorya Golamudi - 191IT252
Information Technology
National Institute of Technology Karnataka
Surathkal, India 575025
Email: sooryahyma.191it252@nitk.edu.in

Niraj Nandish - 191IT234
Information Technology
National Institute of Technology Karnataka
Surathkal, India 575025
Email : nirajn.191it234@nitk.edu.in

***Abstract*—In distributed systems, multiple computers interact with each other over a network to perform data processing operations parallel over a network. Distributed systems can be set up depending on the application, and its data with a different combination of replications and then the partitioning of data can be performed using sharding techniques are, it is difficult to determine which particular shard need to be searched to retrieve the relevant document matching to the user query. Moreover, to rank the documents which have the highest relevance to the user query. The main goal is to develop a mechanism to handle large scale data, processing the searching operations using efficient sharding technique.**

I. Introduction

Sharding, also known as horizontal partitioning, is used when the data set is too large for the database. Sharding permits the respective database to cluster along with the increase in data and traffic growth. It is meant to make an extensive database more manageable. The idea is to distribute data that cannot fit on a single node onto a cluster of database nodes. Sharding can be done both manually and automatically, although the manual method is much more efficient and therefore preferred.

When adding resources to a database, the default method of vertical scaling reaches the point of diminishing, whereas, with sharding, the horizontal partitioning spans out the compute capacity and has faster query time. Sharding can further be very useful in the case of unplanned outages. In conclusion, sharding can increase storage capacity, fasten the processing, and offer broader availability at a lower cost.

SINR (Signal to interference plus noise ratio) is a way to measure the quality of wireless connections or the quantity from which we get the theoretical upper bounds of a channel's rate of information transfer in wireless communication. It is calculated by dividing the power of a particular signal by sum of interference power and background noise power.

Challenges:

1) Uneven storage of data (usually occurs in manual sharding) can cause them to be unbalanced.
2) Too much data on a shard can cause server crashes.

II. Literature Survey

Performance Analysis of Distributed Processing System using Shard Selection Techniques on Elasticsearch

A paper by Praveen M Dhulavvagol, Vijayakumar H Bhajantri, S G Totad that deals with sharding in D2D networks and serves as a base for this paper and project.

NS3 documentation

The NS3 documentation was used for reference code for several aspects of the project.

Using NS-3 as base, we have created an interconnected system using Public Safety Communicated Model, which provides extensions to LTE devices the power of Device to Device (D2D) capabilities.

The model initially developed and maintained by the Public Safety Communication Project Group is based on the ns-3.31 release and supports the following features:

- Out of coverage synchronization
- In and out of coverage D2D Discovery
- In and out of coverage D2D Communication
- Various 3GPP based propagation models
- detailed off-network MCPTT model for D2D operation
- UAV mobility energy model
- HTTP application model
- UE-to-Network Relay model
- Additional examples for multi-cell and partial coverage

The model is used in Large Events, Pandemics, Accidents and Fire and Emergency Medical Services, which gives high efficiency of speed and throughput.

III. Problem Statement

Given D2D and CU, divide the UEs into shards so that each shard contains at least one CU based on criteria like distance or SINR, or resources available. Divide the D2D user equipment into pairs and calculate the SINR of all the pairs.

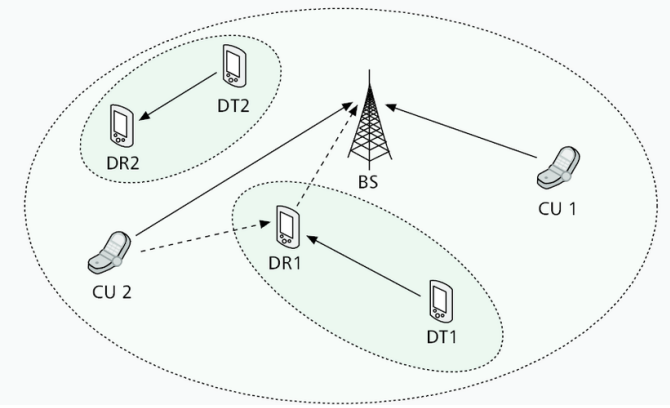

Fig. 1. D2D network Sharding

IV. Methodology

A. K Means Clustering Algorithm

This algorithms partitions n observations into k clusters(distinct non-overlapping subgroups) in which each observation belongs to a cluster with the nearest mean.

K Means Clustering is chosen as it is simple and easy to implement. It also requires centroids and uses features such as the distance of each object to the centroids to cluster the given nodes which could be easily implemented by substituting the CU nodes for centroids and UE nodes as the nodes to be classified.

Algorithm:

1) Specify number of clusters K.
2) Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.
3) Keep iterating until there is no change to the centroids. i.e. assignment of data points to clusters isn't changing.
4) Compute the sum of the squared distance between data points and all centroids.
5) Assign each data point to the closest cluster (centroid).
6) Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

B. Working

We create three types of nodes, namely eNB Node, UE Nodes, also known as D2D nodes and CU nodes which are the cellular user nodes.

In our project, we have created one eNB Node at the centre and 3 CU Nodes and 10 UE Nodes.

We create the nodes using

```
NodeContainer enbNode; // For eNB Node
enbNode.Create(1);


NodeContainer ueNodes; // For UE Node
NodeContainer ueClients;
NodeContainer ueServers;
//Client - Server as it is D2D

for (uint32_t i = 0; i < node_num; ++i)
    {
        NodeContainer pair;
        pair.Create(2);

        ueClients.Add(pair.Get(0));
        ueServers.Add(pair.Get(1));

        ueNodes.Add(pair);

        uePairs[i].Add(pair);
    }

NodeContainer cuNodes; // For CU Node
cuNodes.Create(shard_num);
```

We have represented the number of UE Nodes with node num, which values to 10 and CU Nodes to shard num, which is valued at 3.

Then we allocate random locations (x and y coordinates) to the CU and UE Nodes which will be used by the Sharding Algorithm.

For the eNB Node we have set the location to be (0,0) which will the centre of the whole layout and the other CU and EU nodes have random x and y values ranging from 0 to 100.

We have created a Struct Point which will store the coordinates and IP - Addresses of the Nodes.

```
struct NodeData
{
    double x, y;
    int cluster;
    uint32_t node_ID1;
    uint32_t node_ID2;
    int node_type;
    Ipv4Address node_Ip1;
    Ipv4Address node_Ip2;
    double minDist;

    NodeData(double x, double y, int t,
      uint32_t node1, uint32_t node2 = -1)
```

```
        : x(x), y(y), cluster(-1),
          node_ID1(node1), node_ID2(node2)
           , node_type(t),
             minDist(__DBL_MAX__)
    {
    }
    double distance(NodeData p)
    {
        return (p.x - x) * (p.x - x) +
          (p.y - y) * (p.y - y);
    }
};
```

Newly Added Components to basic K-Means Algorithm:

- x,y - Stores Coordinates Value
- cluster - Stores Shard/Cluster Number
- node ID1 - Stores Node Number for Client
- node ID2 - Stores Node Number for Server
- node type - Stores whether is it UE Node or CU Node (0 for UE and 1 for CU)
- node ip1 - Stores the IP Address of the node 1 in D2D
- node ip1 - Stores the IP Address of the node 2 in D2D

Addition to this, we can add a few more changes in the K-Means Sharding Algorithm. As compared to the K-Means Algorithm where a cluster centroid is chosen as random, we are choosing a specific centroid for each cluster which will be the CU Nodes.

```
for(vector<Point>::iterator
        x = points->begin();
            x!=points->end(); ++x)
{
  if(x->node_type == 1&& x->cluster==-1)
  centroids.push_back(*x);
}
```

Here node type == 1 denotes a CU Node and cluster == -1 denotes it has not been already assigned to a cluster before.

Rest of the K Means Cluster follows the normal Algorithm for Clustering.

We also have a csv file generated from the Algorithm which gives the details of the node with their cluster/shard number.

As compared to the normal NS-3 file , we have included changes to incorporate CU nodes in it at all places.

We have created a Node Container which contains all the nodes and we then install all the nodes to the Internet Stack Helper.

Also to get the IP Addresses of each of the nodes in IPV4 system :

```
for(int i = 0;i<n;i++){
           points[i].node_Ip =ueNodes.Get
           (i)->GetObject<Ipv4L3Protocol>()
           ->GetAddress(1, 0).GetLocal();
   } // For the UE Nodes
for(int i = 0;i<k;i++){
           points[n+i].node_Ip =cuNodes.Get
           (i)->GetObject<Ipv4L3Protocol>()
           ->GetAddress(1, 0).GetLocal();
   } // For the CU Nodes
```

node_type	node_id	node_ip_address	x	y	shard
0	4	7.0.0.2	49	32	1
0	5	7.0.0.3	49	32	1
0	6	7.0.0.4	25	41	0
0	7	7.0.0.5	25	41	0
0	8	7.0.0.6	76	33	2
0	9	7.0.0.7	76	33	2
0	10	7.0.0.8	61	60	2
0	11	7.0.0.9	61	60	2
0	12	7.0.0.10	29	80	1
0	13	7.0.0.11	29	80	1
0	14	7.0.0.12	45	51	1
0	15	7.0.0.13	45	51	1
0	16	7.0.0.14	36	93	2
0	17	7.0.0.15	36	93	2
0	18	7.0.0.16	80	61	2
0	19	7.0.0.17	80	61	2
0	20	7.0.0.18	45	21	1
0	21	7.0.0.19	45	21	1
0	22	7.0.0.20	63	18	1
0	23	7.0.0.21	63	18	1
1	24	7.0.0.22	22	57	0
1	25	7.0.0.23	35	57	1
1	26	7.0.0.24	64	74	2

Fig. 2. CSV File

We then display the shards with the IP addresses of the UE Nodes in the shard with the corresponding CU Node.

C. SINR Calculation

```
//This is the Monitor Called for SINR Output
void monitor(std::string context,
    uint16_t cellId , uint16_t rnti,
     double rsrp, double sinr,
      uint8_t componentCarrierId)
{
    std::cerr << "\n";
    std::cerr << "context:␣" << context << "\n";
```

```
    std::cerr << "sinr" << sinr << "\n";
    std::cerr << "\n";
}
```

We found out that there is a call back function that takes in multiple parameters including SINR. We used the config::connect and gave it a path of

```
/NodeList//DeviceList//
  $ns3::LteUeNetDevice/
    ComponentCarrierMapUe/*/\LteUePhy/
      ReportCurrentCellRsrpSinr
```

to calculate the SINR. Now we used the SINR parameters from this.

```
for (auto &&shard : shards)
{
    for (auto &&point : shard)
    {
    Config::Connect("/NodeList/" +
     to_string(point.node_ID2) +
      "/DeviceList/0/$ns3::LteUeNetDevice
␣␣␣␣␣␣/ComponentCarrierMapUe/*/LteUePhy/
␣␣␣␣␣␣ReportCurrentCellRsrpSinr",
       MakeCallback(&monitor));

        }
    }
```

Serial No.	Seconds	Active D2D Pairs	SINR @ Destination
1	2.5	Node 4 - Node 5	113.5
2	3.5	Node 6 - Node 7	167.789
3	4.5	Node 8 - Node 9	204.859
4	5.5	Node 10 - Node 11	65.1508
5	6.5	Node 12- Node 13	100.752
6	7.5	Node 14 - Node 15	71761.5
7	8.5	Node 16 - Node 17	139.55
8	9.5	Node 18 - Node 19	6296.93
9	10.5	Node 20 - Node 21	90.8939
10	11.5	Node 22 - Node 23	70.1261

Fig. 3. SINR Values for D2D Nodes

V. Results

The output of the cc file is displayed below. The result is that the clustering of the different nodes into shards using K means clustering. The program also calculates the SINR values of the signals received at the destination nodes in the order of receiving. The output is as comprehensive as possible and gives information about the nodes such as their IP addresses, shards they belong to, node ids and the SINR values to help properly understand the simulation and its features. Such a detailed output can help engineers learn a lot about real-life situations and allow them to come up with solutions to cope with difficulties at the source nodes, destinations or in the medium.

```
(base) niraj@LAPTOP-66VR7FFQ:~/psc-ns3-3.0.1$ ./waf --run scratch/CNN_-_Project
Waf: Entering directory `/home/niraj/psc-ns3-3.0.1/build'
[2941/2994] Compiling scratch/CNN_-_Project.cc
[2942/2994] Compiling scratch/scratch-simulator.cc
[2951/2994] Compiling scratch/subdir/scratch-simulator-subdir.cc
[2952/2994] Linking build/scratch/scratch-simulator
[2953/2994] Linking build/scratch/subdir/subdir
[2954/2994] Linking build/scratch/CNN_-_Project
Waf: Leaving directory `/home/niraj/psc-ns3-3.0.1/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (39.612s)
eNb node id = [3]
CU1node id = [24]
CU2node id = [25]
CU3node id = [26]
node_type       node_id         node_ip_address         x       y       shard
0       4       7.0.0.2         84      47      0
0       5       7.0.0.3         84      47      0
0       6       7.0.0.4         22      46      1
0       7       7.0.0.5         22      46      1
0       8       7.0.0.6         47      79      0
0       9       7.0.0.7         47      79      0
0       10      7.0.0.8         32      40      1
0       11      7.0.0.9         32      40      1
0       12      7.0.0.10        90      43      2
0       13      7.0.0.11        90      43      2
0       14      7.0.0.12        62      39      2
0       15      7.0.0.13        62      39      2
0       16      7.0.0.14        63      50      2
0       17      7.0.0.15        63      50      2
0       18      7.0.0.16        5       29      1
0       19      7.0.0.17        5       29      1
0       20      7.0.0.18        52      59      0
0       21      7.0.0.19        52      59      0
0       22      7.0.0.20        56      61      0
0       23      7.0.0.21        56      61      0
1       24      7.0.0.22        68      76      0
1       25      7.0.0.23        14      23      1
1       26      7.0.0.24        56      28      2
```

Fig. 4. D2D Sharding Output in terminal

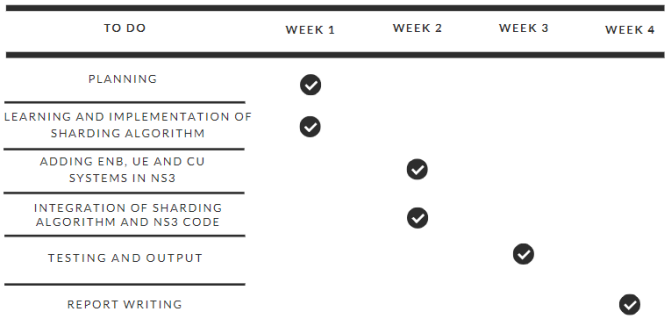

Fig. 5. Gantt Chart

VI. Acknowledgment

The authors would like to acknowledge the support of our guide Mr. Kiran M and we would also like to acknowledge numerous discussions on the subject with him.

VII. Implemented/Base Paper

Name : Performance Analysis of Distributed Processing System using Shard Selection Techniques on Elasticsearch

Author : Praveen M Dhulavvagol, Vijayakumar H Bhajantri, S G Totad

Conferenece : International Conference on Computational Intelligence and Data Science

Published Year : 2019

Authors	Methodology	Merits	Limitations
Praveen M Dhulavvagol, Vijayakumar H Bhajantri, S G Totad	Shard Selection using Elastic Search	Sharding is well implemented and explained	Doesn't use K-means clustering

Fig. 6. Summary of Literature Survey

REFERENCES

[1] NS3 documentation
[2] Data Communications and Networking, Behrouz A. Forouzan, 4th Edition, McGraw Hill, 2017
[3] IT200 Course material